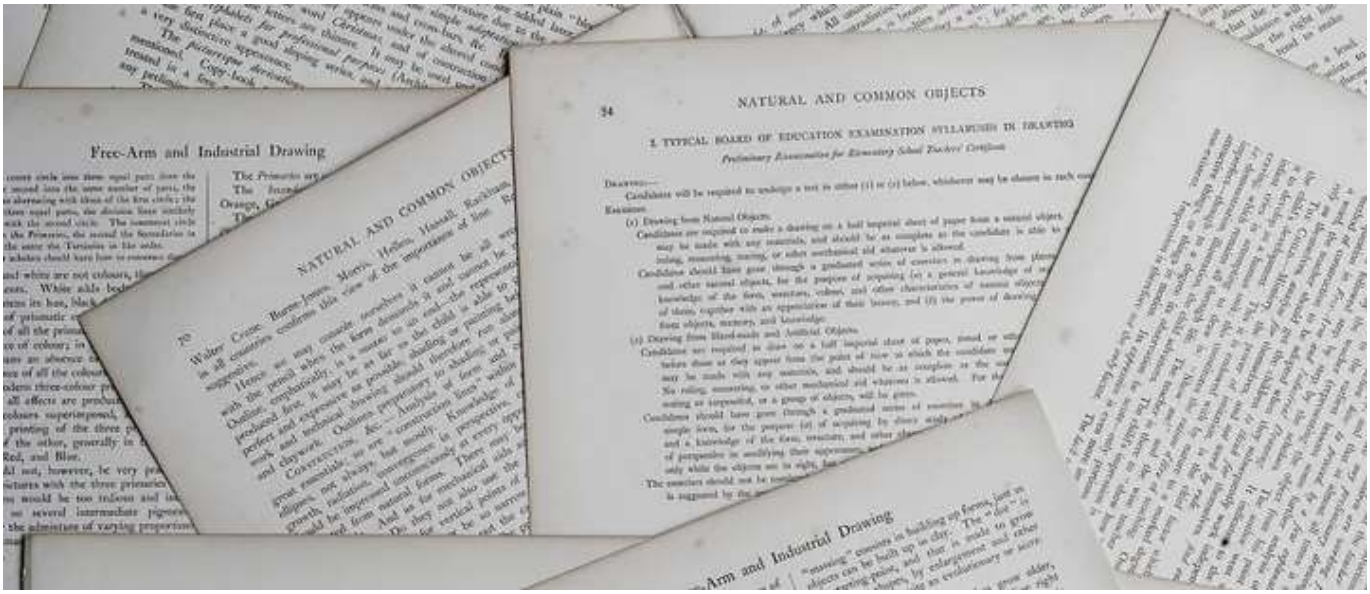
Multi-Page Document Classification | Part-3

Qaisar Tanvir | Lead Data Scientist · Follow

8 min read · Aug 2, 2021

5



This article describes a novel **Multi-Page Document Classification** solution approach, which leverages advanced machine learning and textual analytics to solve one of the major challenges in Mortgage industry. This is the **part 3** of our series of blogs. You can find the links to the different parts of the series below.

- Part 1: **Abstract, Introduction** (*Background and Problem Statement, Objective*), **Characteristics of Documents.**
- Part 2: **Solution Methodology** (*ML Classes, ML Engine, Post-Processing*)
- Part 3: **Solution Details** (*Data Preparation, Data Transformation*), **Training Pipeline** (*Text Vectorizer Doc2Vec, Machine Learning Classifier, Training Procedure*)
- Part 4: **Testing & Evaluation Pipeline, Solution Features, Conclusion**

Solution Details

In this blog, we will have a deep-dive into different components of the solution pipeline. Please refer to previous parts of this series if there are any confusions.

Data Preparation

In pursuit of developing an end-to-end document classification pipeline. the very first, and arguably the most important step is data preparation, because the solution is as good as the data it uses. The data we used for our experiments, were documents from the mortgage domain. The strategies we adopted, can be applied to any form of document datasets in the similar fashion. Following are the steps which were performed.

> **Definition** *: **Document Sample** is an instance of a particular document. Usually it is a (**pdf**) file containing only the pages of that document.*

Step 1

- First step is to decide, *which documents within a package are to be recognized and classified?*. Ideally, all the documents which are present in packages should be selected. Once the document classes are decided we

move onto the extraction part. In our case we decided to classify (44) document classes.

Step 2

- To obtain the data-set, we collected pdfs of several hundred packages, and manually extracted the selected documents from those packages. Once the document was identified in the package, the pages of that document were separated and concatenated together in the form of a pdf file. For example if we had found "**Doc A**" from page 4 to 10 in a package. we would extract the 6 pages (4–10) and merge them into a 6-pager pdf. This 6-pager pdf constitutes a **document sample.** All the samples extracted for a particular document class was put into a seperate folder. Following shows the folder structure. We collected **300+** document samples for each document class. Each document class was given a unique identifier which we called "**DocumentIdentifierID**"

Step 3

- The next step is to apply **OCR** and extract text from all the pages present in the document samples. The OCR iterated on all the folders and generated excel files, having the extract text and some meta-data. Following shows the format of the excel files, Each row represents one page

LoanNumber	DocumentIdentifierID	DocumentName	FileName	PageCount	PageNumber	IsLastPage	PageText
201631809	6271	4506T	201631809_IRSFORM4506T_1.pdf	2	1	0	Form 4506 T Request for
201631809	6271	4506T	201631809_IRSFORM4506T_1.pdf	2	2	1	Form 4506-T (Rev. 7-2017
201757870	6271	4506T	201757870_IRSFORM4506T_1.pdf	2	1	0	DocuSign Envelope ID: F
201757870	6271	4506T	201757870_IRSFORM4506T_1.pdf	2	2	1	DocuSign Envelope ID: F
...	...	...	...	...	...	...	

Dataset Table with sample rows.

Loan Number, File Name : These are unique sample (pdf) identifiers. There are two (green, yellow) samples present in the table.

Document Identifier ID , Document Name : Represent the document class, which these samples belong to.

Page Count : Total number of pages present in one particular sample. (both samples have 2 pages)

Page Number : Is the ordered page number of each page within a sample.

IsLastPage : If 1, it means the page is the last page of that particular sample.

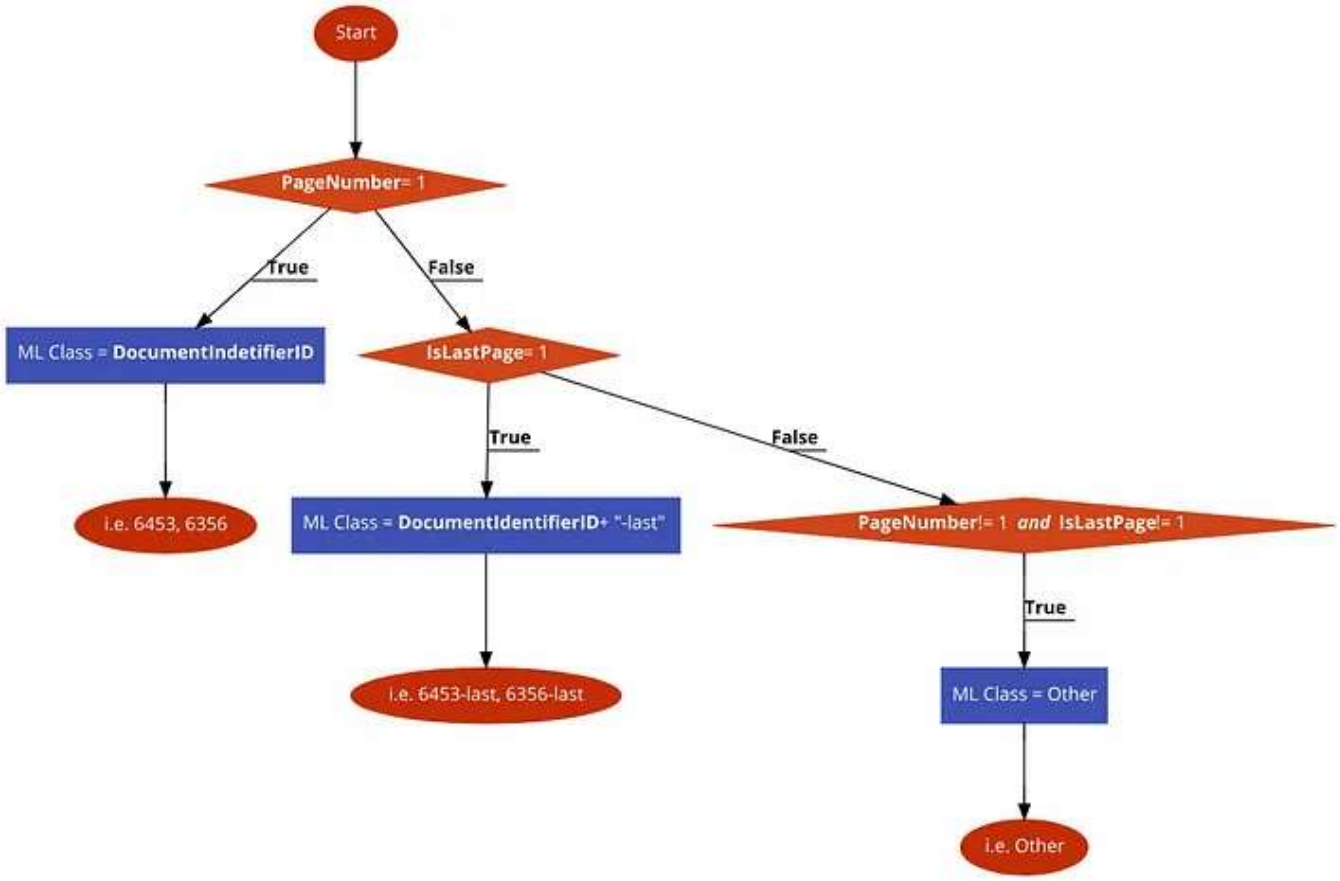
Page Text : Is the text returned from the OCR for that particular page.

Data Transformations

Once the data is generated in the above format, next step is to transform it. In the transformations phase, the data is converted/manipulated into the format which is essential for training a machine learning model. Following are the transformations which are applied to the dataset.

Step 1 | Generating ML classes

- First step of transformation is to generate *first page, last page, and other page classes*. To do this, *Page Number* and *IsLastPage* columns values are used. Following shows a conditional representation of the logic used.



- Moreover, below table represents the columns. Notice the yellow column where **6853** represents the first page class, **6853-last** represents the last page class, while mid-pages are considered as **Other** class

LoanNumber	DocumentIdentifierID	DocumentName	FileName	PageCount	PageNumber	IsLastPage	PageText	ML Class
1064000140	6853	Application Initia	1064000140_201001_Application Ir	4	1	0	Uniform Residen	6853
1064000140	6853	Application Initia	1064000140_201001_Application Ir	4	2	0	LOAN #: 0122025	Other
1064000140	6853	Application Initia	1064000140_201001_Application Ir	4	3	0	LOAN #: 0122025	Other
1064000140	6853	Application Initia	1064000140_201001_Application Ir	4	4	1	LOAN #: 0122025	6853-last
...	...	...	...	...	...	...	...	...

Step 2 | Data Split for Training and Test the Pipeline

- Once the step 1 is complete, from that point on we only need two columns "**Page Text**" and "**ML Class**" to make the training pipeline. Other columns are used for testing evaluations.
- Next step is to split the data for training and testing the pipeline, The data is split in a way where **80%** is used for training and **20%** is used for testing. The data is also randomly shuffled, but in a stratified fashion for each class. For more information click the link.

Step 3 | Data cleaning and transformation

- The "**Page Text**" column which contains the OCR text for each page is cleaned, this process is applied on train and test both. Following are the processes which are performed.

1. **Case correction** : All the text is converted to UPPER or lower case.
2. **Regex for non-alphanumeric characters** : All the characters which are not alphanumeric are removed.
3. **Word Tokenization** : All the words are tokenized, which means the one *Page Text* string becomes list of words
4. **Stopwords Removal** : Stopwords are the words which are too common in the English language and might not be helpful in classifying the individual documents. For example words like "the", "is", "a". These words can also be domain specific. it can be used to remove redundant words, which are common in many different documents. i.e. in terms of finance or mortgage, the word "price" can occur in many documents.

Following tables show before and after transformations

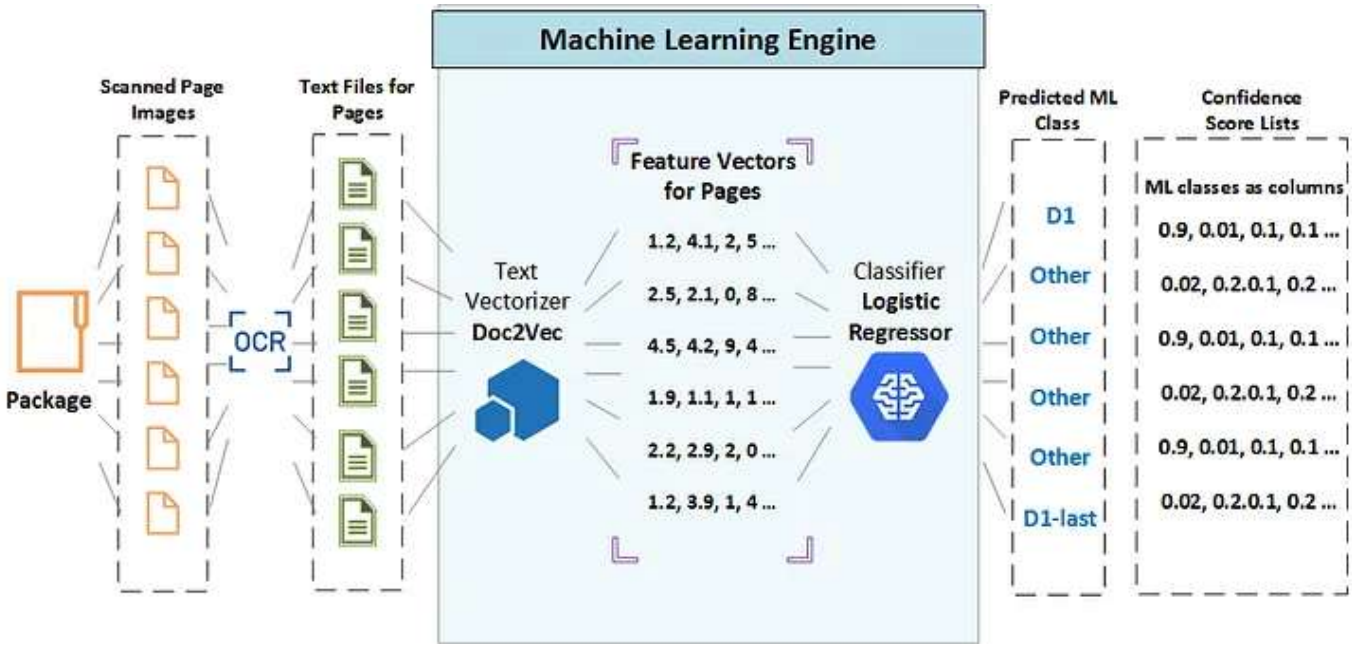
PageText	ML Class
those liabilities which will be satisfied upon sale of real estate owned or upon refinancing	6853
Statement can be meaningfully and fairly presented on a combined basis	Other
visual observation or surname if you have	Other
Federal crime punishable by fine or imprisonment, or both, to knowingly make any	6853-last

After transformations

PageText	ML Class
those', 'liabilities', 'which', 'satisfied', 'upon', 'sale', 'real', 'estate', 'owned', 'upon', 'refinancing'	6853
statement', 'meaningfully', 'fairly', 'presented','combined', 'basis'	Other
visual', 'observation', 'surname'	Other
federal', 'crime', 'punishable', 'fine','imprisonment', 'both', 'knowingly', 'make'	6853-last

Training Pipeline

In the previous **blog,** we abstractly discussed the inner workings of the Machine Learning Engine. For better understanding, let's revisit that diagram and discuss the two main components to the solution.



1. **Text Vectorizer :** In our case we have used **Doc2Vec**
2. **Classifier Model :** Logistic Regressor is used for classification.

Text Vectorizer (Doc2Vec)

Since the beginning of the **Natural Language Processing** (NLP), there has been the need to transform text into something a machine can understand. Which means, transforming textual information into a meaningful representation which is usually known as vectors (or array) of numbers. Research community has been developing different methods to perform this

Doc2Vec is based on **Word2Vec** model. Word2Vec model is a **Predictive Vector Space Model.** To understand Word2Vec, let us begin with **Vector Space Models.**

> ***Vector Space Models (VSMs):*** *Embeds words into a continuous vector space where semantically similar words are mapped to nearby points*

Two Approaches for VSM:

1. ***Count-Based Methods:*** Compute the statistics of how often some word co-occurs with its neighbor words in a large text corpus, and then map these count-statistics down to a small, dense vector for each word (e.g. TFIDF)
2. ***Predictive Methods:*** Predict a word from its neighbors in terms of learned small, dense embedding vectors (e.g. Skip-Gram, CBOW). Word2Vec and Doc2Vec belong to this category of models

Word2Vec Model

It is a computationally efficient predictive model for learning word embedding from raw text. **Word2Vec** can be created by using the following two models:

1. **Skip-Gram:** Creates a sliding window around current word (target word). Then use current word to predict all surrounding words (the context words). (e.g. predicts **'the cat sits on the'** from **'mat'**)
2. **Continuous Bag-of-Words** (**CBOW**): Creates a sliding window around current word (target word). Then predict the current word from surrounding words (the context words). (e.g. predicts **'mat'** from **'the cat sits on the'**)

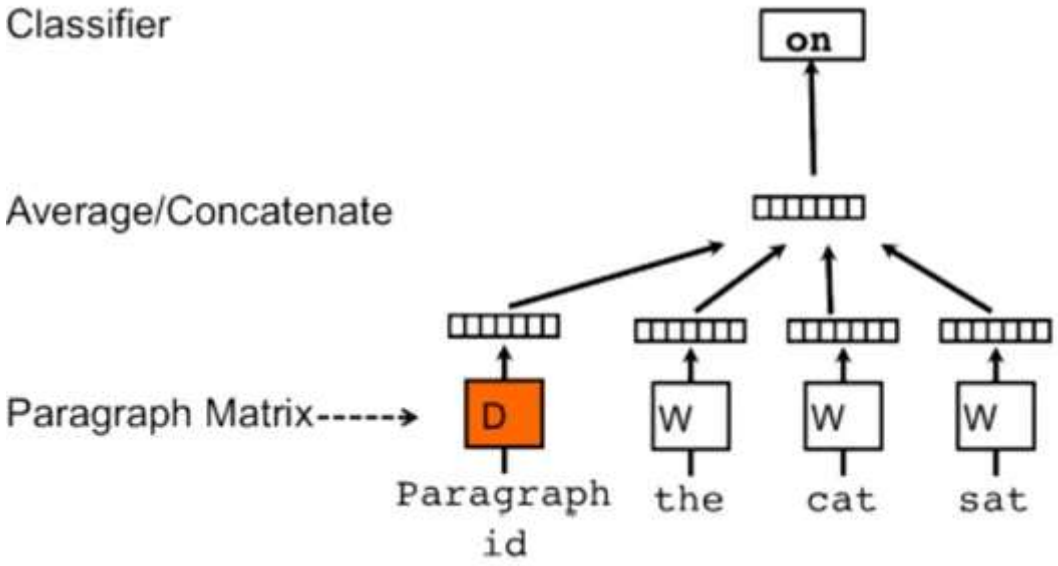
For more details, read this article. it explains different aspects of it in detail.

Doc2Vec Model

This text vectorization technique was introduced in the scientific research paper **Distributed Representations of Sentences and Documents.** Moreover, further technical details can be found here.

Definition | *it is an* ***unsupervised algorithm*** *that learns* ***fixed-length feature vector representation*** *from variable-length pieces of texts. Then these vectors can be used in any machine learning classifier to predict the classes label.*

It is similar to Word2Vec model except, it uses all words in each text file to create a unique column in a matrix (called it **Paragraph Matrix**). Then a single layer NN, like the one seen in Skip-Gram model, will be trained where the input data are all surrounding words of the current word along with the current paragraph column to predict the current word. The rest is same as the Skip-Gram or CBOW models.



Doc2Vec | Distributed Bag of Words | Source: **Distributed Representations of Sentences and Documents**

The advantage of Doc2Vec model:

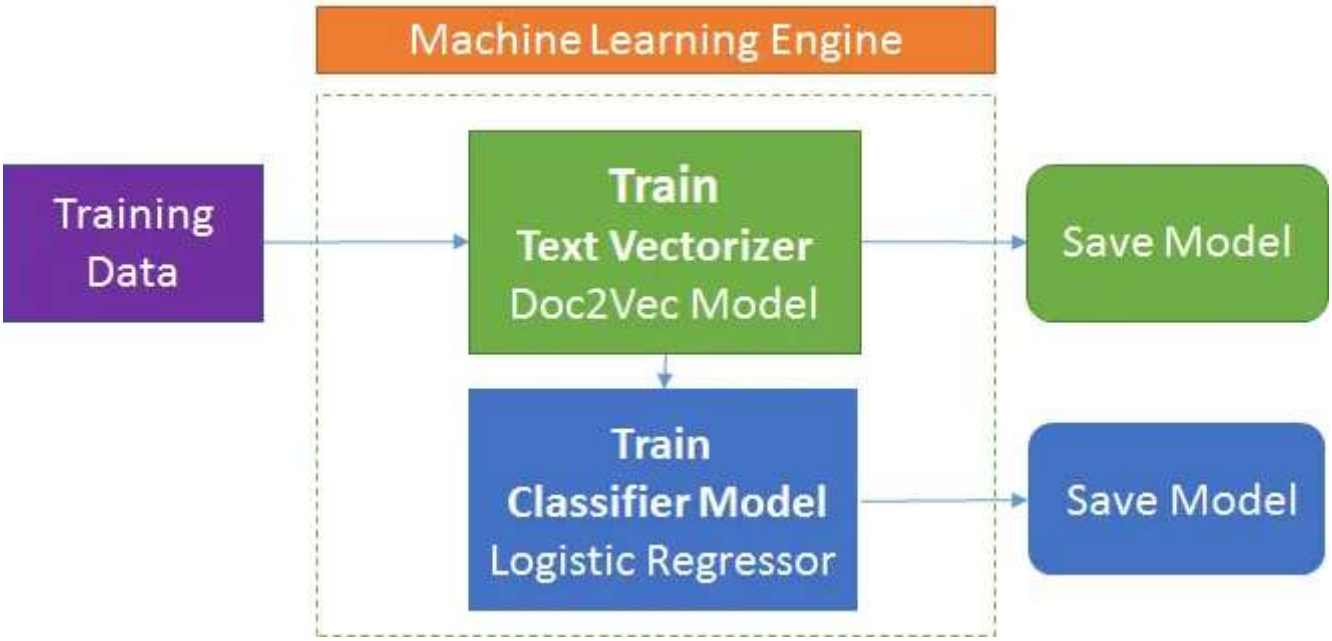
- On **sentiment analysis task**, Doc2Vec achieves new state-of-the-art results, better than complex methods, yielding a relative improvement of more than **16%** in terms of error rate.
- On **text classification task**, Doc2Vec convincingly beats bag-of-words models, giving a relative improvement of about **30%**.

Classifier Model (Logistic Regressor)

Once the text is converted to a vector format. it is ready for a machine learning classifier to learn the patterns present in the vectors of different document types and identify the correct distinctions. Since, there are many classification techniques which can be used here, we tried best of the bunch and evaluated their results. i.e. Random Forest, SVM, Multi-Layer Perceptron and Logistic Regressor. Many different parameters were tried for each classifier to obtain the optimal results. **Logistic Regressor** was found to be the best amongst all of these models.

Training Procedure

- Once the data is transformed. Firstly, we train the Doc2Vec model on the training split (as discussed in the data transformation section). -
- After the Doc2Vec model is trained. the training data is passed through it again, but this time the model is not trained, rather we infer the vectors for the training samples. The last step is to pass these vectors and the actual ML class label to the classification model (Logistic Regressor).
- Once the models are trained on the training data, the both models are saved to the disk, so that these can be loaded into memory to be used in testing and ultimate production deployment. Following diagram shows the basic flow of this collaborative scheme.



In this blog, we have briefly discussed the various steps of our solution pipeline. we discussed the data preparation and data transformation steps in detail. Moreover, the technical components of the **Machine Learning Engine** were discussed in detail. In the next blog we will discuss the testing and evaluation techniques and key things to consider when making a text classification solution. Following is the link.

Next Blog

- Part 4: Testing & Evaluation Pipeline, Solution Features, Conclusion

Machine Learning | NLP | Document Classification | Python | Doc2vec

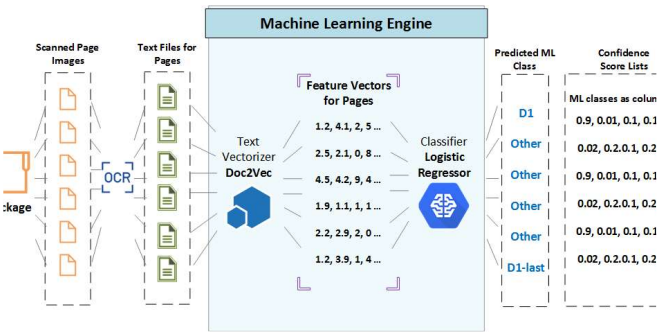
Written by Qaisar Tanvir | Lead Data Scientist

Follow

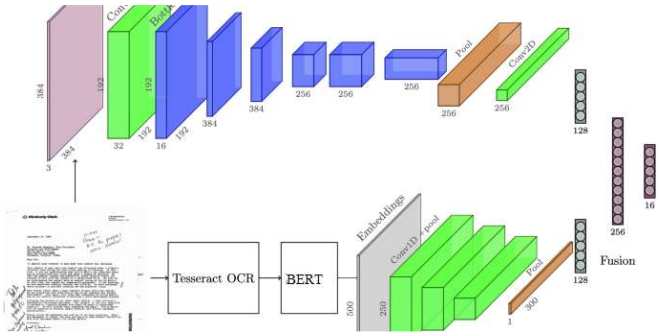
124 Followers

With applied experience in many industries i.e. retail, health, insurance & finance.
linkedin.com/in/qaisartanvir qaisar.tanvir@outlook.com

More from Qaisar Tanvir | Lead Data Scientist



Qaisar Tanvir | Lead Data Sc... in Towards Data Sci...



Qaisar Tanvir | Lead Data Scientist in Towards AI

Multi Page Document Classification using Machine...

An approach to classify documents with different variations shapes, text and page...

19 min read · Aug 7, 2021

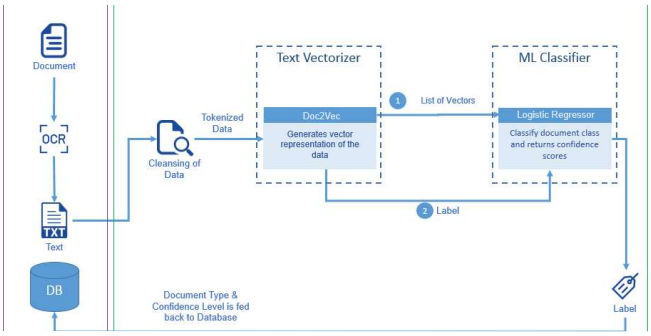
279 11

Multimodal Deep Multipage Document Classification using...

Document AI using python and Tensorflow, using CNN (for image) and BERT (for text),...

8 min read · Mar 24

104



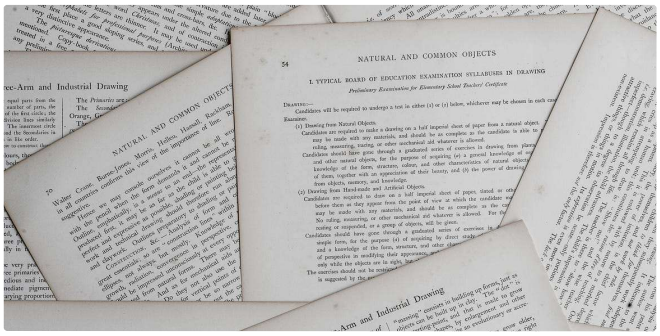
Qaisar Tanvir | Lead Data Scientist

Multi-Page Document Classification | Part-2

This article describes a novel Multi-Page Document Classification solution approach,...

5 min read · Aug 2, 2021

9



Qaisar Tanvir | Lead Data Scientist

Multi-Page Document Classification | Part-1

This article describes a novel Multi-Page Document Classification solution approach,...

5 min read · Aug 2, 2021

12

See all from Qaisar Tanvir | Lead Data Scientist

Recommended from Medium

Khang Pham

Text Classification with BERT

In this tutorial, we will use BERT to develop your own text classification model.

8 min read · May 9

42 1

Abdallah Ashraf

Text Extraction and Clean-up in NLP

Text extraction and cleanup refer to extracting the raw textual content from input...

8 min read · Jul 31

143

Lists

Predictive Modeling w/ Python

20 stories · 506 saves

Practical Guides to Machine Learning

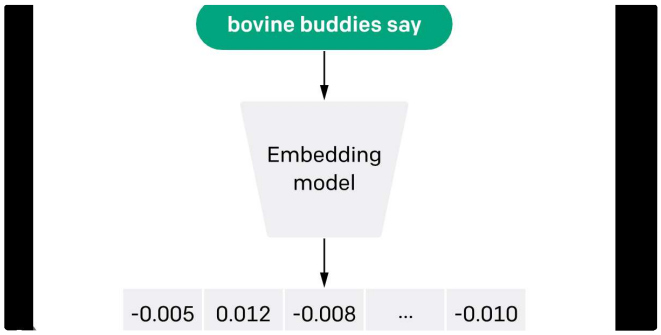
10 stories · 580 saves

Natural Language Processing

734 stories · 324 saves

Coding & Development

11 stories · 223 saves



Tejpal Kumawat

Empowering Natural Language Processing with OpenAI...

Introduction

6 min read · Jun 11

6

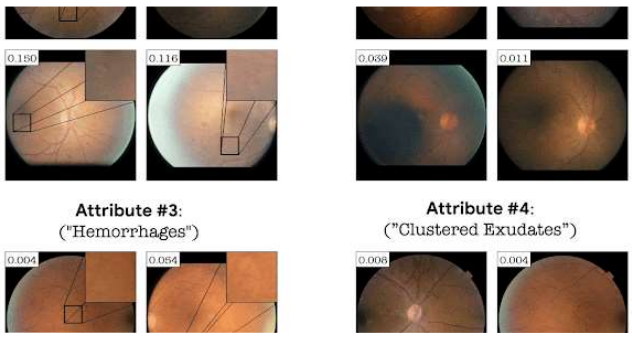
Stefan Neefischer

Semantic keyword clustering in Python

We already shared some clustering approaches using TF-IDF Vectorizer for...

3 min read · May 15

29



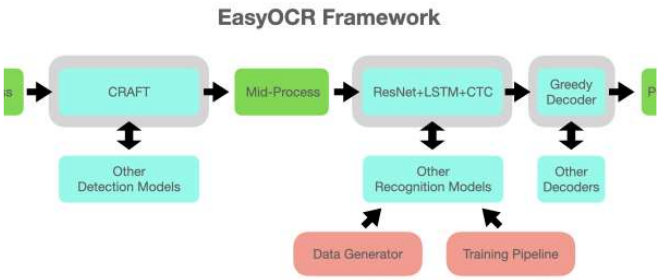
Everton Gomede, PhD

Multi-Label Classification in Python: Empowering Machine...

Introduction

6 min read · May 25

22



Mohamed Hasan

OCR with Deep Learning in PyTorch (EasyOCR)

Part 1: A Beginner Guide of OCR with Python Code

3 min read · May 1

17

See more recommendations